

“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”

“UNIVERSIDAD PERUANA LOS ANDES”

FACULTAD DE INGENIERÍA

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y
COMPUTACIÓN**



Trabajo 3

Alumno:

- Pariona Gozme Maicol Brayan

Docente: Fernandez Bejarano Raul Enrique

Asignatura: ARQUITECTURA DE SOFTWARE

Sección: B1

Ciclo: VII

Huancayo - Perú

2024

Sistema de Ventas al Contado y al Crédito

El presente proyecto consiste en el desarrollo de un sistema de gestión de ventas que permita a los usuarios realizar transacciones tanto al contado como al crédito. El sistema está diseñado para ser utilizado en entornos comerciales, facilitando la gestión de clientes, productos y transacciones, y brindando un seguimiento claro de las ventas realizadas.

Objetivo del Proyecto

El objetivo principal de este sistema es ofrecer una solución eficiente y efectiva para la gestión de ventas, que permita a los vendedores registrar las transacciones de manera rápida y sencilla, ya sea al contado o al crédito. Además, el sistema debe permitir calcular automáticamente subtotales, descuentos y montos mensuales en el caso de las ventas al crédito.

Requerimientos Funcionales

<i>Nombre:</i>	<i>R1: Registrar Venta al Contado</i>
<i>Resumen:</i>	<i>Permite al usuario registrar una venta al contado, ingresando datos del cliente y del producto.</i>
<i>Entrada:</i>	<i>Nombre del cliente, RUC, fecha, hora, producto, cantidad.</i>
<i>Resultados:</i>	<i>Registro de la venta en el sistema, cálculo de subtotal y descuentos, actualización de la vista.</i>

<i>Nombre:</i>	<i>R2: Calcular Subtotal y Descuentos</i>
<i>Resumen:</i>	<i>Calcula el subtotal de la venta al contado y aplica descuentos según reglas predefinidas.</i>
<i>Entrada:</i>	<i>Datos de la venta (cantidad, precio, descuentos).</i>
<i>Resultados:</i>	<i>Subtotal, descuento aplicado y total a pagar</i>

<i>Nombre:</i>	<i>R3: Mostrar Resumen de la Venta</i>
<i>Resumen:</i>	<i>Muestra un resumen de la venta realizada al contado en la interfaz de usuario.</i>
<i>Entrada:</i>	<i>Datos de la venta (subtotal, descuento, total).</i>
<i>Resultados:</i>	<i>Resumen actualizado en la vista de ventas al contado.</i>

Nombre:	<i>R4: Registrar Venta al Crédito</i>
Resumen:	<i>Permite al usuario registrar una venta al crédito, ingresando datos del cliente y del producto.</i>
Entrada:	<i>Nombre del cliente, RUC, fecha, hora, producto, cantidad, letras.</i>
Resultados:	<i>Registro de la venta en el sistema, cálculo de subtotal, descuentos y monto mensual.</i>

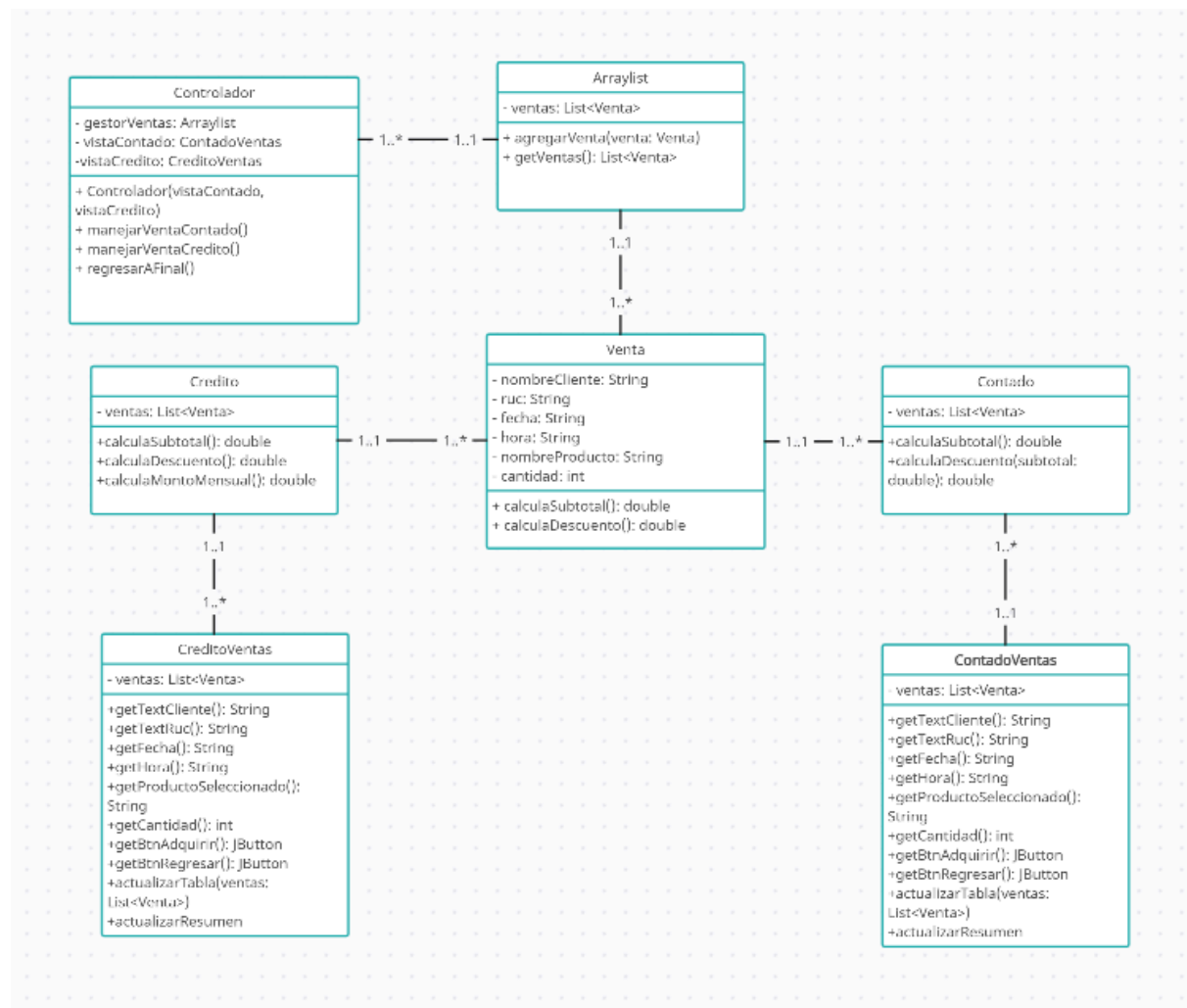
Nombre:	<i>R5: Mostrar Resumen de Venta Crédito</i>
Resumen:	<i>Muestra un resumen de la venta realizada al crédito en la interfaz de usuario.</i>
Entrada:	<i>Datos de la venta (subtotal, descuento, monto mensual).</i>
Resultados:	<i>Resumen actualizado en la vista de ventas al crédito.</i>

Nombre:	<i>R6: Calcular Monto Mensual</i>
Resumen:	<i>Calcula el monto mensual que el cliente debe pagar en caso de una venta al crédito.</i>
Entrada:	<i>Datos de la venta al crédito (subtotal, letras).</i>
Resultados:	<i>Monto mensual calculado.</i>

Nombre:	<i>R7: Actualizar Tabla de Ventas</i>
Resumen:	<i>Actualiza la tabla de ventas en la interfaz de usuario después de registrar una nueva venta.</i>
Entrada:	<i>Nuevas ventas registradas.</i>
Resultados:	<i>Tabla de ventas actualizada en la vista correspondiente</i>

Nombre:	R8: Regresar a Ventana Principal
Resumen:	Permite al usuario regresar a la ventana principal desde las vistas de ventas.
Entrada:	N/A
Resultados:	Muestra la vista de ventas al contado y oculta la vista de ventas al crédito.

Diagrama de Clases



Codigo:

Modelo:

Venta:

```
package modelo;
```

```
/**
```

```
 * Clase base que representa una venta. Contiene métodos para calcular el
```

```
 * subtotal y manejar información del cliente y del producto.
```

```
 */
```

```
public class Venta {
```

```
    private String nombreCliente;
```

```
    private String ruc;
```

```
    private String fecha;
```

```
    private String hora;
```

```
    private String nombreProducto;
```

```
    private int cantidad;
```

```
    private double precio;
```

```
    public Venta(String nombreCliente, String ruc, String fecha, String hora, String nombreProducto, int  
    cantidad) {
```

```
        this.nombreCliente = nombreCliente;
```

```
        this.ruc = ruc;
```

```
        this.fecha = fecha;
```

```
        this.hora = hora;
```

```
        this.nombreProducto = nombreProducto;
```

```
        this.cantidad = cantidad;
```

```
        this.precio = asignaPrecio(nombreProducto);
```

```
    }
```

```
    public static double asignaPrecio(String producto) {
```

```
        switch (producto) {
```

```
            case "Lavadora":
```

```
                return 1500.00;
```

```
            case "Refrigeradora":
```

```
                return 3500.00;
```

```
            case "Licuadora":
```

```
                return 500.00;
```

```
            case "Extractor":
```

```
                return 150.00;
```

```
            case "Radiograbadora":
```

```
                return 750.00;
```

```
            case "DVD":
```

```
                return 100.00;
```

```
            case "Blue Ray":
```

```
                return 250.00;
```

```
            default:
```

```
                return 0.0;
```

```
        }
```

```

    }

    public double calculaSubtotal() {
        return this.precio * this.cantidad;
    }

    public String getNombreCliente() {
        return nombreCliente;
    }

    public String getRuc() {
        return ruc;
    }

    public String getFecha() {
        return fecha;
    }

    public String getHora() {
        return hora;
    }

    public String getNombreProducto() {
        return nombreProducto;
    }

    public int getCantidad() {
        return cantidad;
    }

    public double getPrecio() {
        return precio;
    }

    @Override
    public String toString() {
        return "Venta{" +
            "nombreCliente=" + nombreCliente + "\" +
            ", ruc=" + ruc + "\" +
            ", fecha=" + fecha + "\" +
            ", hora=" + hora + "\" +
            ", nombreProducto=" + nombreProducto + "\" +
            ", cantidad=" + cantidad +
            ", precio=" + precio +
            "'";
    }
}

```

Credito:

```
package modelo;
```

```
public class Credito extends Venta {
    private int letras; // Número de letras para el pago
    private int x; // Cantidad de productos comprados

    // Constructor
    public Credito(String nombreCliente, String ruc, String fecha, String hora, String nombreProducto,
int cantidad, int letras) {
        super(nombreCliente, ruc, fecha, hora, nombreProducto, cantidad); // Llama al constructor de la
clase base
        this.letras = letras; // Asigna el número de letras
        this.x = cantidad; // Asigna la cantidad de productos
    }

    // Método para obtener la cantidad de productos
    public int getX() {
        return x;
    }

    // Métodos getter y setter para letras
    public int getLetras() {
        return letras;
    }

    public void setLetras(int letras) {
        this.letras = letras;
    }

    // Método para calcular el descuento basado en el subtotal
    public double calculaDescuento() {
        double subtotal = calculaSubtotal(); // Utiliza el método de la clase base
        if (subtotal < 1000.00) {
            return subtotal * 0.03; // 3% de descuento
        } else if (subtotal >= 1000.00 && subtotal <= 3000.00) {
            return subtotal * 0.05; // 5% de descuento
        } else {
            return subtotal * 0.08; // 8% de descuento
        }
    }

    // Método para calcular el monto mensual a pagar
    public double calculaMontoMensual() {
        double totalPagar = calculaSubtotal() - calculaDescuento(); // Total a pagar después del
descuento
        return letras > 0 ? totalPagar / letras : 0.0; // Si hay letras, calcula el monto mensual
    }

    @Override
    public String toString() {
```

```

        return super.toString() + ", Letras: " + letras + ", Cantidad: " + x;
    }
}

```

Contado:

```

package modelo;

public class Contado extends Venta {
    private int n; // Cantidad de productos

    // Constructor que inicializa la cantidad de productos
    public Contado(String nombreCliente, String ruc, String fecha, String hora, String nombreProducto,
int cantidad) {
        super(nombreCliente, ruc, fecha, hora, nombreProducto, cantidad);
        this.n = cantidad; // Asigna la cantidad de productos comprados
    }

    // Método para obtener la cantidad de productos
    public int getN() {
        return this.n;
    }

    // Método para calcular el descuento basado en el subtotal
    public double calculaDescuento(double subtotal) {
        if (subtotal < 1000.00) {
            return subtotal * 0.05; // 5% de descuento
        } else if (subtotal >= 1000.00 && subtotal <= 3000.00) {
            return subtotal * 0.08; // 8% de descuento
        } else {
            return subtotal * 0.12; // 12% de descuento
        }
    }

    public double calculaDescuento() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }
}

```

Arraylist:

```

package modelo;

import java.util.ArrayList;

public class Arraylist {
    private ArrayList<Venta> ventas;

    public Arraylist() {
        ventas = new ArrayList<>();
    }
}

```



```

    }

    public void agregarVenta(Venta venta) {
        ventas.add(venta);
    }

    public void eliminarVenta(int index) {
        if (index >= 0 && index < ventas.size()) {
            ventas.remove(index);
        } else {
            System.out.println("Índice fuera de rango.");
        }
    }

    public Venta obtenerVenta(int index) {
        if (index >= 0 && index < ventas.size()) {
            return ventas.get(index);
        } else {
            System.out.println("Índice fuera de rango.");
            return null;
        }
    }

    public ArrayList<Venta> getVentas() {
        return ventas;
    }

    public void listarVentas() {
        if (ventas.isEmpty()) {
            System.out.println("No hay ventas registradas.");
        } else {
            for (int i = 0; i < ventas.size(); i++) {
                System.out.println((i + 1) + ". " + ventas.get(i).toString());
            }
        }
    }

    public void buscarVentasPorCliente(String nombreCliente) {
        boolean found = false;
        for (Venta venta : ventas) {
            if (venta.getNombreCliente().toLowerCase().contains(nombreCliente.toLowerCase())) {
                System.out.println(venta.toString());
                found = true;
            }
        }
        if (!found) {
            System.out.println("No se encontraron ventas para el cliente: " + nombreCliente);
        }
    }

```

```

// Método para listar solo ventas al contado
public void listarVentasContado() {

```

```

        for (Venta venta : ventas) {
            if (venta instanceof Contado) {
                System.out.println(venta);
            }
        }
    }
}

// Método para listar solo ventas al crédito
public void listarVentasCredito() {
    for (Venta venta : ventas) {
        if (venta instanceof Credito) {
            System.out.println(venta);
        }
    }
}
}

```

Controlador:

```

package Controlador;

import modelo.Arraylist;
import modelo.Contado;
import modelo.Credito;
import Vista.ContadoVentas;
import Vista.CreditoVentas;
import javax.swing.JOptionPane;

public class Controlador {

    public static void manejarAdquirir() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }

    private Arraylist gestorVentas; // Modelo para almacenar ventas
    private ContadoVentas vistaContado; // Vista para ventas al contado
    private CreditoVentas vistaCredito; // Vista para ventas al crédito

    // Constructor del controlador
    public Controlador(ContadoVentas vistaContado, CreditoVentas vistaCredito) {
        this.gestorVentas = new Arraylist();
        this.vistaContado = vistaContado;
        this.vistaCredito = vistaCredito;

        // Agregar listeners a los botones de las vistas
        this.vistaContado.getBtnAdquirir().addActionListener(e -> manejarVentaContado());
        this.vistaCredito.getBtnAdquirir().addActionListener(e -> manejarVentaCredito());
        this.vistaContado.getBtnRegresar().addActionListener(e -> regresarAFinal());
        this.vistaCredito.getBtnRegresar().addActionListener(e -> regresarAFinal());
    }

    private void manejarVentaContado() {

```

```

// Obtener los datos de la vista
String nombreCliente = vistaContado.getTextCliente();
String ruc = vistaContado.getTextRuc();
String fecha = vistaContado.getFecha();
String hora = vistaContado.getHora();
String nombreProducto = vistaContado.getProductoSeleccionado();
int cantidad = vistaContado.getCantidad();

// Validar datos
if (nombreCliente.isEmpty() || ruc.isEmpty() || cantidad <= 0) {
    JOptionPane.showMessageDialog(vistaContado, "Por favor, completa todos los campos correctamente.", "Error", JOptionPane.ERROR_MESSAGE);
    return;
}

// Crear la venta al contado
Contado ventaContado = new Contado(nombreCliente, ruc, fecha, hora, nombreProducto, cantidad);
gestorVentas.agregarVenta(ventaContado);

// Calcular subtotal y descuento
double subtotal = ventaContado.calculaSubtotal();
double descuento = ventaContado.calculaDescuento(subtotal);

// Actualizar la vista (tabla, resumen, etc.)
vistaContado.actualizarTabla(gestorVentas.getVentas());
vistaContado.actualizarResumen(nombreCliente, ruc, fecha, hora, subtotal, descuento, subtotal - descuento);
}

private void manejarVentaCredito() {
    // Obtener los datos de la vista
    String nombreCliente = vistaCredito.getTextCliente();
    String ruc = vistaCredito.getTextRuc();
    String fecha = vistaCredito.getFecha();
    String hora = vistaCredito.getHora();
    String nombreProducto = vistaCredito.getProductoSeleccionado();
    int cantidad = vistaCredito.getCantidad();
    int letras = vistaCredito.getLetras(); // Obtener el número de letras

    // Validar datos
    if (nombreCliente.isEmpty() || ruc.isEmpty() || cantidad <= 0 || letras <= 0) {
        JOptionPane.showMessageDialog(vistaCredito, "Por favor, completa todos los campos correctamente.", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Crear la venta al crédito
    Credito ventaCredito = new Credito(nombreCliente, ruc, fecha, hora, nombreProducto, cantidad, letras);
    gestorVentas.agregarVenta(ventaCredito);
}

```

```

// Calcular subtotal, descuento y monto mensual
double subtotal = ventaCredito.calculaSubtotal();
double descuento = ventaCredito.calculaDescuento();
double montoMensual = ventaCredito.calculaMontoMensual();

// Actualizar la vista (tabla, resumen, etc.)
vistaCredito.actualizarTabla(gestorVentas.getVentas());
}

// Método para regresar a la ventana principal
private void regresarAFinal() {
    vistaContado.setVisible(true); // Mostrar la vista de ventas al contado
    vistaCredito.setVisible(false); // Ocultar la vista de ventas al crédito
}
}

```

Vista

package Vista;

```

import modelo.ArrayList;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Timer;
import java.util.TimerTask;
import modelo.Contado;
import modelo.Venta;

public class ContadoVentas extends javax.swing.JFrame {
    private ArrayList gestorVentas; // Instancia del gestor de ventas
    private VentasInicial ventasInicial; // Referencia a la ventana principal

    public ContadoVentas(VentasInicial ventasInicial) {
        initComponents();
        this.setTitle("Contado Ventas");
        this.setLocationRelativeTo(null);
        this.ventasInicial = ventasInicial; // Guardar la referencia de la ventana principal
        gestorVentas = new ArrayList(); // Inicializa el gestor de ventas

        // Actualizar la hora inicialmente
        actualizarHora();

        // Usar un Timer para actualizar la hora cada segundo
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {

```

```

        actualizarHora();
    }
}, 0, 1000); // Actualiza cada 1000 milisegundos (1 segundo)
}

// Método para actualizar la hora en jLabelHora
private void actualizarHora() {
    SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
    String horaActual = dateFormat.format(new java.util.Date());
    jLabelHora.setText(horaActual); // Actualiza la etiqueta con la hora actual
}

// Método para obtener el botón de adquirir
public javax.swing.JButton getBtnAdquirir() {
    return btnAdquirir; // Devuelve la referencia al botón de adquirir
}

public JButton getBtnRegresar() {
    return btnRegresar; // Ensure this returns the correct button reference
}

// Métodos Get para obtener datos de la vista
public String getTextCliente() {
    return jTextCliente.getText(); // Devuelve el texto del campo cliente
}

public String getTextRuc() {
    return jTextRuc.getText(); // Devuelve el texto del campo RUC
}

public String getFecha() {
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    return dateFormat.format(jDateCalendario.getDate()); // Devuelve la fecha seleccionada
}

public String getHora() {
    return jLabelHora.getText(); // Devuelve la hora actual
}

public String getProductoSeleccionado() {
    return (String) jComboBoxProducto.getSelectedItem(); // Devuelve el producto seleccionado
}

public int getCantidad() {
    return Integer.parseInt(jTextFieldCantidad.getText()); // Devuelve la cantidad ingresada
}

private void jTextClienteActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jTextRucActionPerformed(java.awt.event.ActionEvent evt) {

}

```

```

private void jTextFieldCantidadActionPerformed(java.awt.event.ActionEvent evt) {

}

// Método para calcular el descuento basado en el subtotal
private double calcularDescuento(double subtotal) {
    if (subtotal < 1000) {
        return subtotal * 0.05; // 5% de descuento
    } else if (subtotal >= 1000 && subtotal <= 3000) {
        return subtotal * 0.08; // 8% de descuento
    } else {
        return subtotal * 0.12; // 12% de descuento
    }
}

// Método para actualizar la tabla
public void actualizarTabla(ArrayList<Venta> ventas) {
    DefaultTableModel model = (DefaultTableModel) jTableCredito.getModel();
    model.setRowCount(0); // Limpiar tabla antes de agregar nuevas filas

    for (Venta venta : ventas) {
        model.addRow(new Object[]{
            // Agrega aquí los datos de la venta que deseas mostrar
            venta.getNombreProducto(),
            venta.getCantidad(),
            String.format("%.2f", venta.calculaSubtotal())
        });
    }
}

private void btnAdquirirActionPerformed(java.awt.event.ActionEvent evt) {
    // Obtener los datos de los campos de texto
    String nombreCliente = jTextFieldCliente.getText();

    // Validar que el nombre del cliente no esté vacío
    if (nombreCliente.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Por favor, ingresa un nombre de cliente.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Validar RUC
    String ruc = jTextFieldRuc.getText();
    if (!ruc.matches("\\d+")) { // Verifica que el RUC contenga solo dígitos
        JOptionPane.showMessageDialog(this, "Por favor, ingresa un RUC válido (solo números).",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Obtener fecha y hora
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    String fecha = dateFormat.format(jDateCalendario.getDate()); // Obtener fecha del calendario
    String hora = jLabelHora.getText(); // Obtener la hora actual desde jLabelHora
    String nombreProducto = (String) jComboBoxProducto.getSelectedItem();

```

```

// Validar que se haya seleccionado un producto
if (nombreProducto == null) {
    JOptionPane.showMessageDialog(this, "Por favor, selecciona un producto.", "Error",
JOptionPane.ERROR_MESSAGE);
    return;
}

int cantidad;

// Validar cantidad
try {
    cantidad = Integer.parseInt(jTextFieldCantidad.getText()); // Convierte la cantidad a entero
    if (cantidad <= 0) {
        throw new NumberFormatException(); // Lanza excepción si la cantidad no es positiva
    }
} catch (NumberFormatException e) {
    JOptionPane.showMessageDialog(this, "Por favor, ingresa una cantidad válida (solo números).",
"Error", JOptionPane.ERROR_MESSAGE);
    return; // Salir si hay un error de conversión
}

// Obtener el precio del producto
double precio = obtenerPrecioPorProducto(nombreProducto); // Método para obtener el precio del
producto

// Calcular subtotal
double subtotal = cantidad * precio;

// Calcular descuento basado en el subtotal
double descuento = calcularDescuento(subtotal);

// Calcular el neto a pagar
double netoPagar = subtotal - descuento;

// Agregar los datos a la tabla
DefaultTableModel model = (DefaultTableModel) jTableCredito.getModel();

// Contar el número de filas actuales para el ítem
int itemCount = model.getRowCount() + 1; // El ID será 1 más el número actual de filas

// Agregar la fila a la tabla
model.addRow(new Object[]{
    itemCount,           // Ítem (ID secuencial)
    nombreProducto,      // Descripción del producto
    cantidad,            // Cantidad
    String.format("$%.2f", precio), // Precio
    String.format("$%.2f", subtotal) // Subtotal
});

// Actualizar el resumen en jLabelResumen
actualizarResumen(nombreCliente, ruc, fecha, hora, subtotal, descuento, netoPagar);

```

```

// Actualizar el JLabel `pagoneto` para mostrar el neto a pagar
pagoneto.setText(String.format("Neto a Pagar: $%.2f", netoPagar));

// Limpiar los campos de entrada después de agregar la venta
jTextCliente.setText("");
jTextRuc.setText("");
jTextFieldCantidad.setText("");
}

// Método para actualizar el resumen en jLabelResumen
// Método para actualizar el resumen
public void actualizarResumen(String nombreCliente, String ruc, String fecha, String hora, double
subtotal, double descuento, double netoPagar) {
    // Crear resumen de la venta
    String resumenVenta = "<html><b>**RESUMEN DE VENTA**</b><br>" +
        "-----<br>" +
        "CLIENTE: " + nombreCliente + "<br>" +
        "RUC: " + ruc + "<br>" +
        "FECHA: " + fecha + "<br>" +
        "HORA: " + hora + "<br>" +
        "-----<br>" +
        String.format("SUBTOTAL: $%.2f<br>", subtotal) +
        String.format("DESCUENTO: $%.2f<br>", descuento) +
        String.format("NETO A PAGAR: $%.2f", netoPagar) +
        "</html>";

    // Actualizar el jLabelResumen con el resumen formateado
    jLabelResumen.setText(resumenVenta); // Usa HTML para el formato de texto
}

private void btnRegresarActionPerformed(java.awt.event.ActionEvent evt) {
    // Hacer visible la ventana principal y cerrar la ventana actual
    ventasInicial.setVisible(true); // Mostrar la ventana principal
    this.dispose(); // Cerrar la ventana actual
}

private void jComboBoxProductoActionPerformed(java.awt.event.ActionEvent evt) {
}

// Método para obtener el precio del producto
private double obtenerPrecioPorProducto(String producto) {
    switch (producto) {
        case "Lavadora":
            return 1500.00;
        case "Refrigeradora":
            return 3500.00;
        case "Licuadora":
            return 500.00;
        case "Extractor":
            return 150.00;
        case "Radiograbadora":
            return 750.00;
        case "DVD":

```



```

        return 100.00;
    case "Blue Ray":
        return 250.00;
    default:
        return 0.0; // Producto no encontrado
    }
}

```

CreditoVentas

```
package Vista;
```

```

import modelo.Arraylist;
import modelo.Credito;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Timer;
import java.util.TimerTask;
import modelo.Venta;

```

```
/**
```

```
*
```

```
* @author USER 17
```

```
*/
```

```

public class CreditoVentas extends javax.swing.JFrame {
    private ArrayList gestorVentas; // Instancia del gestor de ventas
    private VentasInicial ventasInicial; // Referencia a la ventana principal

```

```
    // Constructor que recibe la ventana principal
```

```

    public CreditoVentas(VentasInicial ventasInicial) {
        initComponents();
        initComponents();
        this.setTitle("Credito Ventas");
        this.setLocationRelativeTo(null);
        this.ventasInicial = ventasInicial; // Guardar la referencia de la ventana principal
        gestorVentas = new ArrayList(); // Inicializa el gestor de ventas
        // Actualizar la hora inicialmente
        actualizarHora();

```

```
    // Usar un Timer para actualizar la hora cada segundo
```

```

    Timer timer = new Timer();
    timer.scheduleAtFixedRate(new TimerTask() {
        @Override
        public void run() {
            actualizarHora();
        }
    }, 0, 1000); // Actualiza cada 1000 milisegundos (1 segundo)
}

```

```
private void actualizarHora() {
```

```

        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
        String horaActual = dateFormat.format(new java.util.Date());
        jLabelHora.setText(horaActual); // Actualiza la etiqueta con la hora actual
    }

    public JButton getBtnAdquirir() {
        return btnAdquirirCredito; // Devuelve la referencia al botón de adquirir
    }

    public JButton getBtnRegresar() {
        return btnRegresar; // Ensure this returns the correct button reference
    }

    // Método para actualizar la tabla
    public void actualizarTabla(ArrayList<Venta> ventas) {
        DefaultTableModel model = (DefaultTableModel) jTableCredito.getModel();
        model.setRowCount(0); // Limpiar tabla antes de agregar nuevas filas

        for (Venta venta : ventas) {
            model.addRow(new Object[]{
                venta.getNombreProducto(),
                venta.getCantidad(),
                String.format("$%.2f", venta.calculaSubtotal())
            });
        }
    }

    // Obtener el nombre del cliente
    public String getTextCliente() {
        return jTextFieldCliente.getText();
    }

    // Obtener el RUC
    public String getTextRuc() {
        return jTextFieldRuc.getText();
    }

    // Obtener la fecha
    public String getFecha() {
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
        return dateFormat.format(jDateCalendario.getDate());
    }

    // Obtener la hora
    public String getHora() {
        return jLabelHora.getText();
    }

    // Obtener el producto seleccionado
    public String getProductoSeleccionado() {
        return (String) jComboBoxProducto.getSelectedItem();
    }

    // Obtener la cantidad

```

```

public int getCantidad() {
    return Integer.parseInt(jTextFieldCantidad.getText());
}

// Obtener el número de letras
public int getLetras() {
    return Integer.parseInt((String) jComboBoxLetras.getSelectedItem());
}

```

VentasInicial

```

package Vista;

/**
 *
 * @author USER 17
 */
public class VentasInicial extends javax.swing.JFrame {

    /**
     * Creates new form VentasInicial
     */
    public VentasInicial() {
        initComponents();
        this.setTitle("Venta Inicial");
        this.setLocationRelativeTo(null);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        btnCredito = new javax.swing.JButton();
        btnContado = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Segoe UI", 3, 36)); // NOI18N
        jLabel1.setText("VENTA DE PRODUCTOS");

        btnCredito.setText("VENTA AL CREDITO");
        btnCredito.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnCreditoActionPerformed(evt);
            }
        });
    }

```

```

    }
});

btnContado.setText("VENTA AL CONTADO");
btnContado.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnContadoActionPerformed(evt);
    }
});

jLabel2.setText("VENTA AL CREDITO");

jLabel3.setText("VENTA AL CONTADO");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(71, 71, 71)
            .addComponent(jLabel2)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel3)
            .addGap(84, 84, 84))
        .addGroup(layout.createSequentialGroup()
            .addGap(27, 27, 27)
            .addComponent(btnCredito, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(btnContado, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(44, 44, 44))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(69, Short.MAX_VALUE)
            .addComponent(jLabel1)
            .addGap(93, 93, 93))
    );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(26, 26, 26)
            .addComponent(jLabel1)
            .addGap(41, 41, 41)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(btnCredito, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(btnContado, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

```

```

        .addComponent(jLabel2)
        .addComponent(jLabel3))
        .addContainerGap(87, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void btnCreditoActionPerformed(java.awt.event.ActionEvent evt) {
    // Crear e iniciar la ventana de ventas a crédito
    CreditoVentas creditoVentas = new CreditoVentas(this); // Pasar la referencia de la ventana
principal
    creditoVentas.setVisible(true); // Hacer visible la ventana de crédito
    this.setVisible(false); // Ocultar la ventana principal
}

private void btnContadoActionPerformed(java.awt.event.ActionEvent evt) {
    // Crear e iniciar la ventana de ventas al contado
    ContadoVentas contadoVentas = new ContadoVentas(this); // Pasar la referencia de la ventana
principal
    contadoVentas.setVisible(true); // Hacer visible la ventana de contado
    this.setVisible(false); // Ocultar la ventana principal
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(VentasInicial.class.getName()).log(java.util.logging.Level.SEVERE,
        null, ex);
    } catch (InstantiationException ex) {

        java.util.logging.Logger.getLogger(VentasInicial.class.getName()).log(java.util.logging.Level.SEVERE,
        null, ex);
    } catch (IllegalAccessException ex) {

```

```
java.util.logging.Logger.getLogger(VentasInicial.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(VentasInicial.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
//</editor-fold>
```

```
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VentasInicial().setVisible(true);
        }
    });
}
```

```
// Variables declaration - do not modify
private javax.swing.JButton btnContado;
private javax.swing.JButton btnCredito;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
// End of variables declaration
}
```

```
package com.mycompany.venta_de_productos;
```

```
import Vista.VentasInicial;
```

```
/**
 *
 * @author USER 17
 */
```

```
public class VENTA_DE_PRODUCTOS {
```

```
    public static void main(String args[]) {
```

```
        try {
            // Set HiFi Look and Feel from JTattoo
            javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.hifi.HiFiLookAndFeel");
        } catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(VENTA_DE_PRODUCTOS.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(VENTA_DE_PRODUCTOS.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
```

```

    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VENTA_DE_PRODUCTOS.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VENTA_DE_PRODUCTOS.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VentasInicial().setVisible(true);
        }
    });
}
}

```